**Allowing employees to submit for overtime**

When allowing an employee to submit for overtime we must check first if they are not already working the overtime based on their shift code. i.e. an employee who works on the ‘A’ crew should not be allowed to submit for overtime where ‘A’ crew is already working.

The shift crew working this date and shiftnumber is ‘A’

This means we need a way to calculate what shift code is going to be working any given date and shift number.

* Create function that takes a Date and a Shift Number
* {
  + Get the difference between the date given and a base date(ex Jan 20 2016)
  + Modulo this difference in days by 28(DaysIntoRotation) – This will give a number between 0 and 27 corresponding to a specific day into the shift code rotation
  + If DaysIntoRotation is less than 7
  + {
    - If the Shift Number given is 1
    - {
      * The shift crew working this date and shiftnumber is ‘A’
    - }
    - Else if the shift number given is 2
    - {
      * If DaysIntoRotation modulo 7 is equal to 0
      * {
        + The shift crew working this date and shiftnumber is ‘C’
      * }
      * Else it is ‘B’
    - }
    - Else
    - {
      * If DaysIntoRotation modulo 7 < 2
      * {
        + The shift crew working this date and shift number is ‘D’
      * }
      * Else it is ‘C’
    - }
  + }
  + Else If DaysIntoRotation is less or equal to 13 and greater than 6
  + {
    - If the Shift Number given is 1
    - {
      * The shift crew working this date and shiftnumber is ‘D’
    - }
    - Else if the shift number given is 2
    - {
      * If DaysIntoRotation modulo 7 is equal to 0
      * {
        + The shift crew working this date and shiftnumber is ‘B’
      * }
      * Else it is ‘A’
    - }
    - Else
    - {
      * If DaysIntoRotation modulo 7 < 2
      * {
        + The shift crew working this date and shift number is ‘C’
      * }
      * Else it is ‘B’
    - }
  + }
  + Else If DaysIntoRotation is less than or equal to 20 and greater than 13
  + {
    - If the Shift Number given is 1
    - {
      * The shift crew working this date and shiftnumber is ‘C’
    - }
    - Else if the shift number given is 2
    - {
      * If DaysIntoRotation modulo 7 is equal to 0
      * {
        + The shift crew working this date and shiftnumber is ‘A’
      * }
      * Else it is ‘D’
    - }
    - Else
    - {
      * If DaysIntoRotation modulo 7 < 2
      * {
        + The shift crew working this date and shift number is ‘B’
      * }
      * Else it is ‘A’
    - }
  + }
  + Else
  + {
    - If the Shift Number given is 1
    - {
      * The shift crew working this date and shiftnumber is ‘B’
    - }
    - Else if the shift number given is 2
    - {
      * If DaysIntoRotation modulo 7 is equal to 0
      * {
        + The shift crew working this date and shiftnumber is ‘D’
      * }
      * Else it is ‘C’
    - }
    - Else
    - {
      * If DaysIntoRotation modulo 7 < 2
      * {
        + The shift crew working this date and shift number is ‘A’
      * }
      * Else it is ‘D’
    - }
  + }
* }

You will also have to check the employees job code to make sure they can apply for the overtime.

**Awarding overtime to employees.**

When choosing an employee to award overtime all employees that would have been more eligible must have opportunity overtime hours added to their account. This must be done based on the shift the overtime applies to. Not each individual overtime need being evaluated. When awarding we’ll have to evaluate every employee that shares the same job code and has the ability to work the overtime.

This script would have to run either every 8 hrs or every 24 hours. Preferrably every 8hrs so it runs for each shift. Windows scheduler/oracle scheduler/cron jobs

* Get the overtime needs that need to be evauluated
* Create empty array to hold employees to add opothours to at the end.(ChargedEmployees)
* Iterate through the needs
* {
  + Calulate the shift code using the function above using the needs date and shift number
  + Check shiftcode on the shift before the overtime needs shift. – do this with the previously used function but subtract 1 from shift number.
  + If shift number equals 1
  + {
    - Shift number equals 4
    - Date equals Date minus 1
  + }
  + Use the function for calculating shift with ShiftNumber minus 1 and Date
  + If employee shift code is equal to calculated shift code before the overtime shift then employee is off-going
  + Get all the employees who could have worked this need using job and shift code and order them by othoursworked+opothours and deptseniority. This way they are already sorted by most eligible.(EligibleEmployees)
  + Get the oncomingSubmissions order by othoursworked+opothours and deptseniority. Make sure they are not already awarded.
  + Get the offGoingSubmissions order by othoursworked+opothours and deptseniority. Make sure they are not already awarded
  + Iterate through the array(offGoingSubmissions)
  + {
    - Get the employee who submitted the submission
    - Iterate through overtime needs
    - {
      * Check if employee is working any of them
    - }
    - If employee is already working an overtime need
    - {
      * Remove submission from array
    - }
    - Else
    - {
      * Store this submission and employee
      * Get out of iteration
    - }
  + }
  + If submission to award was not found
  + {
    - Iterate through the array(oncomingSubmissions)
    - {
      * Get the employee who submitted the submission
      * Iterate through overtime needs
      * {
        + Check if employee is working any of them
      * }
      * If employee is already working an overtime need
      * {
        + Remove submission from array
      * }
      * Else
      * {
        + Store this submission and employee
        + Get out of iteration
      * }
    - }
  + }
  + Update need with the most eligible submissionID
  + Update submission with the awarded flag to true
  + Update employees OTHoursWorked field
  + Iterate through EligibleEmployees
  + {
    - If employee is not the employee who was awarded overtime
    - {
      * Add employee id to ChargedEmployees array if not already there
      * Pair this entry with a number to indicate which otblock they should be charged for. (first 4, full 8, last 4)
      * \*\*This doesn’t work\*\*
    - }
  + }
* }
* Iterate through ChargedEmployees array
* {
  + Add opothours for however much they missed
* }

**Finding employees for Unknown Vacancies**

When a supervisor has an unknown vacancy they will create an overtimeneed with the software thus generating a list of eligible employees for supervisor

* Get shift code using previously mentioned method
* Get every employee who meets job code and shift code order by othoursworked+opothours and deptseniority.
* Get overtime needs that have been award ---where submissionID is null
* Create array for employees working first or last 4(blockedEmployees)
* Iterate through employees
* {
  + //If employee is working before or after this shift for more than 4 hours they must be //removed from the array
  + Iterate through overtimeneeds
  + {
    - If employee id is equal to the overtimeneed’s submission’s empid
    - {
      * If the overtime need’s otblock is 2
      * {
        + Remove employee from array
      * }
      * Else
      * {
        + If employee ID is not in blockedEmployees
        + {

Push employee ID into blockedEmployees

* + - * + }
        + Else
        + {

Remove employee from array

* + - * + }
      * }
    - }
  + }
* }

The list is now of all eligible employees off-going sorted by most eligible.

Once a list has been generated for a supervisor the client side will be resposible for sending back the employee that is to be awarded the overtime. Need to keep track of the overtimeneed ID as well. Once you have the employee who will be working the overtime the following will take place.

* Using the Employee ID and the Overtime need ID insert a new submission meeting the requirements of the overtime need with the employees id
* Pull this submission and then update the overtimeneed with the submission id.
* Update the employees OTHoursWorked field

**Overtime Forced onto employees**

When a supervisor cannot find an employee to work an unknown vacancy they have to force an employee to work it.

* Use jobcode and shiftcode to get all employees on the off-going shift who could stay and work the overtime
* Get employees and sort by LastForced and DeptSeniority
* This list is now sorted and contains employees who are off going, correct jobcode, last forced, and deptseniority
* Present the list to the supervisor

Once the supervisor has the list they will approach each employee beginning with the first in the list and inform them that they are being forced to stay over. The employee can choose to refuse ,however, If they are out of refusals the supervisor will be able to tell and then mandate that the employee stays for overtime. If the employee does refuse the follow procedure happens

* The button will send the employees ID
* Use the ID to update the employees ForcedRefusals field with a -1
* Remake list without this employee

Once the supervisor has found an employee who has not refused to work the overtime the following procedure will take place

* The button will send the employees ID
* Use the ID to update the employees ForcedOTHours and the numForced fields and a timestamp for the lastforced field

**DB structure so far**

The eoss db will use four tables; employee,submission,supervisor,and overtimeneed. Employees will have a one-to-many to the submission table and supervisor will have a one-to-many to the overtimeneed table. Employees will make submissions and supervisors will generate overtime needs. Once the time comes for an overtimeneed to be evaluated the most eligible submissions ID will be used to fill the fk column of the overtimeneed table.

Everything is not null except the submission FK in the overtimeneed table.

The employee table holds all employee information. Most of it is overtime hours data of all different types. It also holds the password and email used to login.

--employee table—

Create table employee(

Id int --this will be a globally unique identifier PK

Firstname varchar

Lastname varchar

Password varchar

Phone char

Email vachar --this should also be unique UK

Shiftcode char

Jobcode char

Othoursworked int --this is the actual amount of OT hours an employee worked

Opothours int --this is the opportunity hours accumulated

Forcedothours int --hours and employee has been forced

Forcedrefusals Int --This is how many forced refusals an employee has left

Grantedrefusals int --if an employee is granted a refusal

Numforced int --how many times forced

Numrefused int --how many times refused forced

Nummandated int --mandated is when you are forced without the option of refusing

LastForced timestamp –Keep track of who was forced last

Lastlogin timestamp

Deptseniority int

)

The submission table holds information about an employee who has submitted for overtime. When they want to work, what jobcode, overtime block desired etc..

--submission table

Create table submission(

Id int PK

Empid int FK to employee.id

Submissiondate date --details about the submission from here on

Shift int

Jobcode char

Empcomment text --an employee can have a preferred position

Otblock int --I chose int for just otblock 1,2 or 3(First 4, Full 8, Last 4)

Tstamp timestamp

Awarded boolean --need to know if submission was awarded

)

--supervisor table

Create table supervisor(

Id int PK

Firstname varchar

Lastname varchar

Password varchar

Phone char

Email varchar

lastlogin timestamp

)

A supervisor can generated an overtime need. The overtimeneed table stores this to be seen by employees who wish to submit for overtime. The submissionid FK is left null until a submission has been awarded to work a specific overtime need.

--overtimeneed table

Create table overtimeneed(

Id int PK

Supid int FK to supervisor.id

Otdate date

Shift int

Jobcode char

Otblock int

Tstamp timestamp

Submissionid int FK to submission.id can be null

)